Hierarchical Clustering

Unsupervised Learning

Agenda

- Hierarchical Clustering
  - Distance Matrix
  - Linkage Methods
  - Dendrogram
- Dimension Reduction
- Principal Component Analysis (PCA)
  - Procedure
  - Terminologies
  - Selecting Principal Components

In the previous session...

- In the last session, we studied the K-means algorithm to cluster the dataset
- K-means is a simple algorithm to understand and implement. However, this algorithm has several flaws
- We need to provide the number of clusters (K) to the algorithm, which is not always easy to determine
- The initial centroid assignment affects the formation of final clusters
- The algorithm is sensitive to the presence of outliers

In this session...

- In this session, we study two more clustering techniques: Hierarchical clustering and DBSCAN
- For these two algorithms, there is no need to pre-specify the number of clusters
- Hierarchical clustering is useful when there is a hierarchical structure in the original data
- DBSCAN identifies the noise/ outliers in the data
- The DBSCAN algorithm can form clusters of any arbitrary shape

Hierarchical Clustering

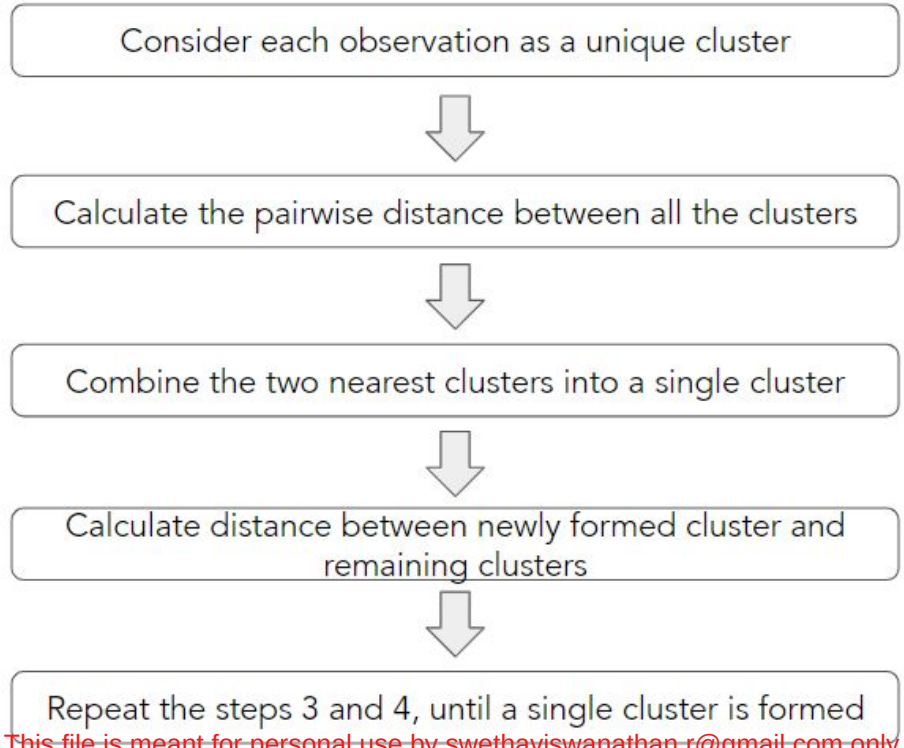
Hierarchical clustering

- Hierarchy based clustering method
- Two main types: Agglomerative (bottom to top approach) and Divisive (top to bottom approach)
- No need to pre-define the number of clusters

Agglomerative clustering

- Most popular hierarchical clustering method
- It considers the bottom to top approach
- The similar observations are clustered together to form a bigger cluster, considering each observation as a unique cluster in the initial step
- The process continues till all the observations are fused in a single cluster
- A dendrogram is used to visualize such cluster formation

Agglomerative clustering - procedure



Great Learning
POWER AHEAD

Python function

In python, the AgglomerativeClustering() performs Agglomerative clustering on the data.

```
# import the function
from sklearn.cluster import AgglomerativeClustering

# pass the number of required clusters
model = AgglomerativeClustering(n_clusters = 2)

# fit and predict the cluster labels
model.fit_predict(data)
```

Distance Matrix

Distance matrix

- In the initial step of clustering, each observation is considered as a cluster
- The distance matrix returns the pairwise distance between all these observations
- The pairwise distance can be calculated using various distance measures like Manhattan, Euclidean, Minkowski and so on
- This matrix is used to find the two closest clusters

Great Learning
POWER AHEAD

Question:

Find the distance matrix for the given data using Euclidean distance.

	X	Y
A	0.1	0.4
B	0.25	0.32
C	0.29	0.19

Answer:

To obtain a distance matrix, calculate the Euclidean distance between all the points.

$$Distance[A, B] = \sqrt[2]{(x-a)^2 + (y-b)^2}$$

$$= \sqrt[2]{(0.1-0.25)^2 + (0.4-0.32)^2}$$

$$= \sqrt[2]{0.0225 + 0.0064}$$

$$= \sqrt[2]{0.0289}$$

$$= 0.17$$

	X	Y
A	0.1	0.4
B	0.25	0.32
C	0.29	0.19

The distance between data points A and B is 0.17. Similarly we can calculate the distance between all the points.

Great Learning
POWER AHEAD

Answer:

By calculating all the distances, we obtain the distance matrix. This matrix returns all the pairwise distances.

	A	B	C
A	0	0.17	0.28
B	0.17	0	0.14
C	0.28	0.14	0

Distance matrix

- The distance of a point from itself will always be zero
- Thus, the diagonal of the distance matrix will be zero
- Here, we have 3x3 matrix for 3 clusters, as each point is considered as a cluster
- We have to group these clusters such that at the end we are left with a single cluster that consists of all the observations

	A	B	C
A	0	0.17	0.28
B	0.17	0	0.14
C	0.28	0.14	0

Distance matrix

- In each iteration, our goal is to find the closest pair
- Here we can see that the distance between the points B and C is the minimum
- Thus, we group B and C in one cluster (B, C)

	A	B	C
A	0	0.17	0.28
B	0.17	0	0.14
C	0.28	0.14	0

Distance matrix

- We will have to update the distances in the distance matrix
- The distance between the ungrouped clusters/elements will remain the same
- But, how we will calculate the distance between the ungrouped clusters and the newly created cluster (B, C)?
- Here is where the different linkage methods are used

Linkage Methods

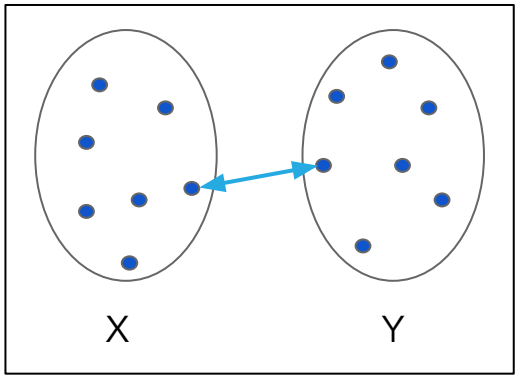
Linkage methods

- Similarity between the clusters (inter-cluster distance) can be measured using various types of linkages
- Some of the types are: Single, Complete, Average, Centroid linkage

Single linkage

It is defined as the minimum distance between the points of two clusters.

$$d(X,Y) = min\{d(x,y)|x\epsilon X, y\epsilon Y\}$$



- The method can create the non-elliptical clusters
- It can produce undesirable results in the presence of outliers
- It causes a chaining effect, where clusters have merged since at least one point in a cluster is closest to a point in another cluster. This forms a long and elongated cluster

Single linkage

In our example, the single linkage between the cluster (B, C) and A is

$$Min[dist(B, A), dist(C, A)]$$
$$= Min[0.17, 0.28]$$
$$= 0.17$$

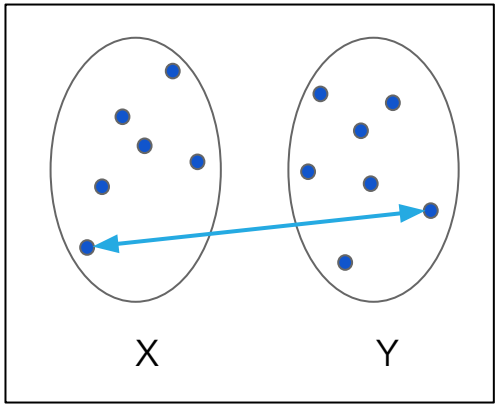
	A	(B, C)
A	0	**?**
(B, C)	**?**	0

	A	(B, C)
A	0	**0.17**
(B, C)	**0.17**	0

Complete linkage

It is defined as the maximum distance between the points of the two different clusters.

$$d(X,Y) = max\{d(x,y)|x\epsilon X, y\epsilon Y\}$$



- The method returns more stable clusters, with nearly equal diameter
- It avoids the chaining effect
- It is less sensitive to outliers
- It breaks the large clusters and it is biased towards globular clusters

Complete linkage

The complete linkage between the cluster (B, C) and A is

$$Max[dist(B, A), dist(C, A)]$$
$$= Max[0.17, 0.28]$$
$$= 0.28$$

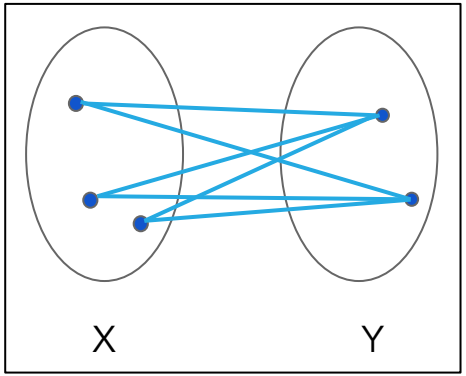
	A	(B, C)
A	0	**?**
(B, C)	**?**	0

	A	(B, C)
A	0	**0.28**
(B, C)	**0.28**	0

Average linkage

It is defined as the average of all the pairwise distances between the two clusters.

$$d(X,Y) = \frac{1}{n_X n_Y} \sum_{x \epsilon X, y \epsilon Y} d(x,y)$$



Where,

n_X : Number of elements in the cluster X

n_Y : Number of elements in the cluster Y

- This method balances between the single and complete linkage
- It forms compact clusters and the method is robust to outliers

Average linkage

The average linkage between the cluster (B, C) and A is

$$AVG[dist(B, A), dist(C, A)]$$
$$= \frac{1}{2}[dist(B, A) + dist(C, A)]$$
$$= \frac{1}{2}[0.17 + 0.28]$$
$$= 0.225$$

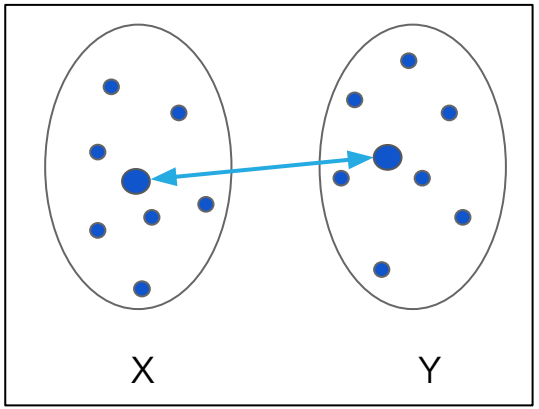
	A	(B, C)
A	0	**?**
(B, C)	**?**	0

	A	(B, C)
A	0	**0.225**
(B, C)	**0.225**	0

Centroid linkage

It is defined as the distance between the centroids (means) of the two clusters.

$$d(X, Y) = d(X_c, Y_c)$$



Where,

X_c : Centroid of the cluster X

Y_c : Centroid of the cluster Y

- It creates similar clusters as average linkage
- This method suffers a major drawback of inversion. i.e. a smaller cluster can be more similar to the newly merged larger cluster rather than the individual clusters

Centroid linkage

The centroid linkage between the cluster (B, C) and A is

$$dist(centroid(A), centroid(B, C))$$
$$= dist((0.1, 0.4), (0.27, 0.255))$$
$$= \sqrt{(0.1 - 0.27)^2 + (0.4 - 0.255)^2}$$
$$= 0.223$$

	A	(B, C)
A	0	**?**
(B, C)	**?**	0

	A	(B, C)
A	0	**0.225**
(B, C)	**0.225**	0

Ward Linkage (ward minimum variance method)

- By default, the scikit-learn library of python considers the 'ward' linkage
- The clusters are merged; if the new cluster minimizes the variance
- It is a computationally intensive method
- It is given by the formula:

$$d(X,Y) = \sqrt{\frac{2.n_X.n_Y}{n_X+n_Y}}.d(X_c,Y_c)$$

X_c : Centroid of the cluster X
Y_c : Centroid of the cluster Y

and

n_X : Number of elements in the cluster X
n_Y : Number of elements in the cluster Y

Summary

Single Linkage

- Can create non-elliptical clusters
- Sensitive to outliers
- Prone to chaining effect

Complete Linkage

- Creates more compact clusters
- Biased towards globular clusters
- Breaks large clusters

Average Linkage

- Balances between single and complete linkage
- Robust to outliers
- Biased towards globular clusters

Centroid Linkage

- Cluster formation is similar to average linkage
- Can cause inversion

Ward Linkage

- Most effective in presence of outliers
- Biased towards globular clusters

Great Learning
POWER AHEAD

Python function

In python, the linkage() returns the linkage matrix. It provides the distance between the cluster given the linkage method.

```
# import the function
from scipy.cluster.hierarchy import linkage

# instantiate linkage object with data & linkage method
# 'link' returns the linkage matrix
link = linkage(df_data, method = 'single')

# print the linkage matrix
print(link)
```

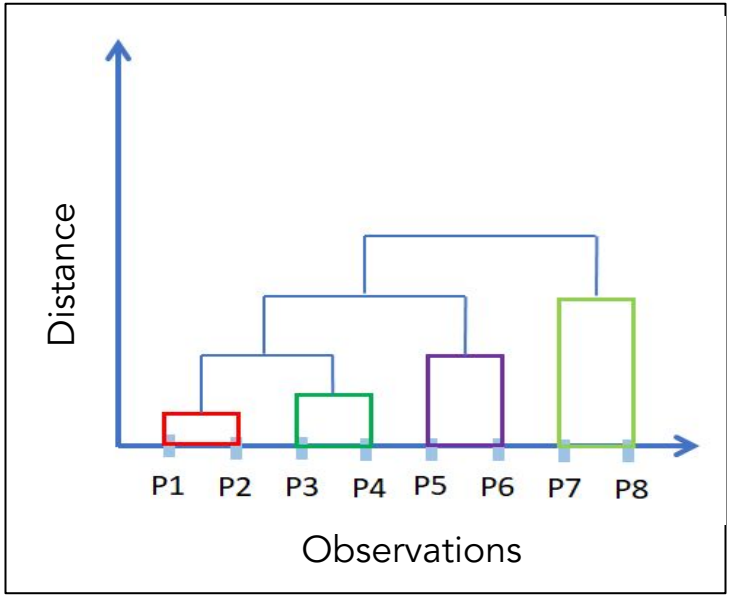
Dendrogram

Dendrogram

- It is a very useful technique to visualize the clusters
- It is a tree-based hierarchical structure that can be used to decide the required number of clusters
- Different linkage methods result in the formation of different dendrograms
- Observations linked at a low height represents more similar observations
- Dissimilar observations fuse at a higher level in the dendrogram

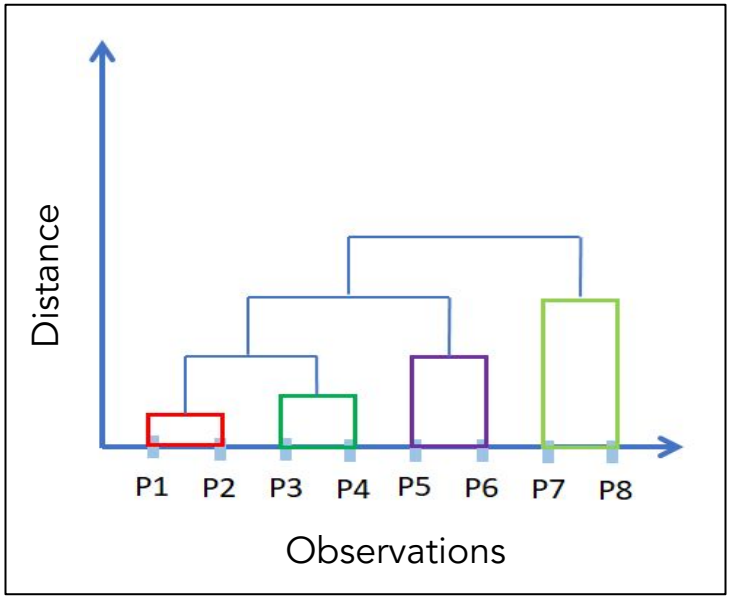
Dendrogram

- X-axis of the dendrogram represents the data point, each considered as a single cluster and the distance is given on the Y-axis
- Each single cluster is known as ‘leaf’
- The horizontal line is known as ‘clade’ which represents the merging of clusters



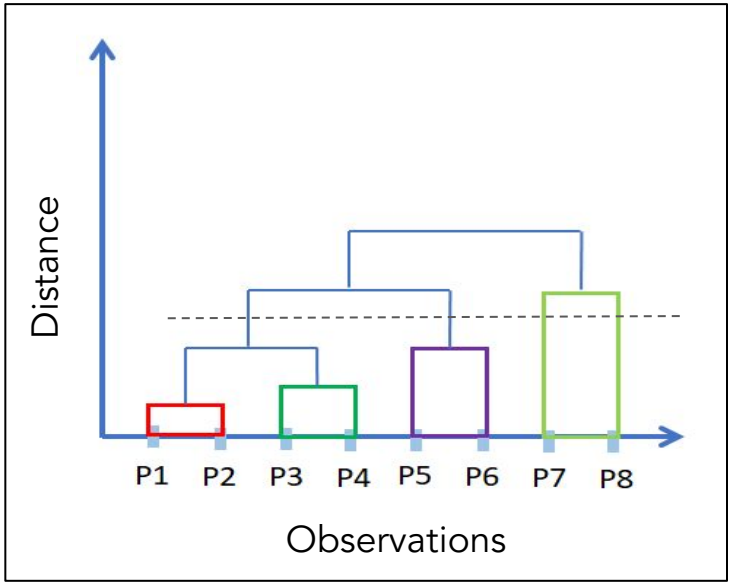
Dendrogram

- P1 and P2 are clustered at the lowest height, which implies more similarity between the observations
- The clusters (P1, P2) and (P3, P4) are clustered to form a bigger cluster. Then this cluster is fused with (P5, P6) to form a even bigger cluster
- Finally, this cluster is fused with (P7, P8) to form a single cluster



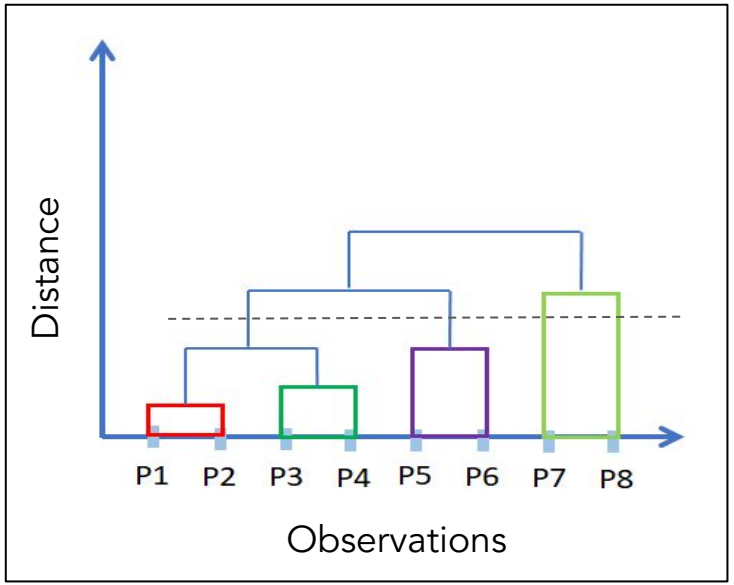
Cutting the dendrogram

- The number of clusters depends on the height at which the dendrogram is cut
- Cutting the dendrogram at different heights results in the formation of distinct clusters
- The optimal number of clusters is the number that remains constant for the larger distance on the y-axis



Dendrogram

- The black line shows the height at which the dendrogram is being cut
- This line intersects the dendrogram at 4 distinct points, which gives 4 clusters namely (P1, P2, P3, P4), (P5, P6), P7, and P8
- The clusters P7 and P8 are clustered at higher distance than the remaining observations, which suggests more dissimilarity between these points



Cophenetic correlation coefficient

- Quantifies how the dendrogram has represented the dissimilarities between the observations
- It is defined as the correlation coefficient between cophenetic distances and the actual distance between the observations
- The cophenetic distance between the points P_i and P_j is the height represented on the Y-axis of dendrogram at which P_i and P_j are first linked together
- The actual distance is the pairwise distance between the observations represented in the distance matrix

Cophenetic correlation coefficient

- The value close to 1 represents the best linkage quality
- It is mostly used in biostatistics to evaluate the cluster models
- In python, the linkage matrix provides the cophenetic distance

```
# import the function
from scipy.cluster.hierarchy import cophenet

# pass the linkage matrix and actual distance
# 1st output of the cophenet() is the correlation coefficient
coeff, cophnet_dist = cophenet(linkage_matrix, actual_dist)

# print the cophnetic correlation coefficient
print(coeff)
```

Great Learning
POWER AHEAD

Python function

In python, the dendrogram() plots the dendrogram for the given linkage matrix.

```
# import the function
from scipy.cluster.hierarchy import dendrogram

# plot the dendrogram
# pass the linkage matrix
dendrogram(linkage_matrix)

# display the plot
plt.show()
```

Summary

- Merits:
  - Does not require a pre-specified number of clusters
  - Hierarchical relation between the clusters can be identified
  - Dendrogram provides a clear representation of clusters
- Demerits:
  - Different dendrograms are produced for different linkage methods
  - Selecting an optimal number of clusters using dendrogram is sometimes difficult
  - Time complexity is high

Great Learning
POWER AHEAD

Silhouette score

- One of the ways to decide the number of clusters is the silhouette score
- We calculate the silhouette score for different values of K (similar to K-means)
- The value of K with the highest silhouette score can be considered as an optimal number of clusters

Case Study

Case study: group the data

Consider the data of flower's petal length and petal width in millimeters for different flowers.

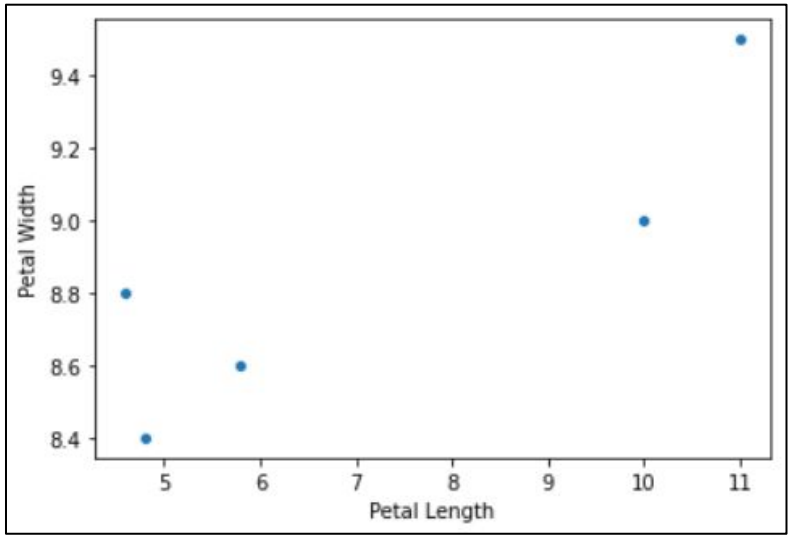
Petal Length	4.8	11	5.8	10	4.6
Petal Width	8.4	9.5	8.6	9	8.8

Can we group the data that belongs to the same kind of flower?

Case study

Use the Euclidean distance as a proximity measure to calculate the distance between the data points.

Use the ‘single’ linkage method to calculate the distance between the two clusters.



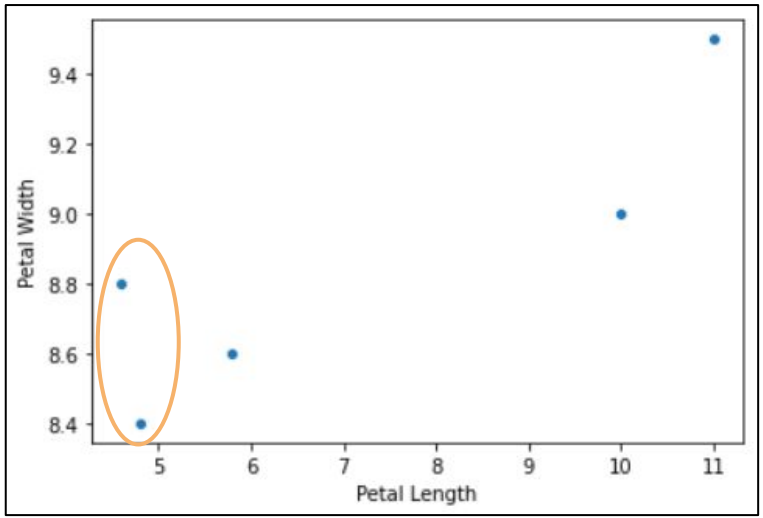
Case study

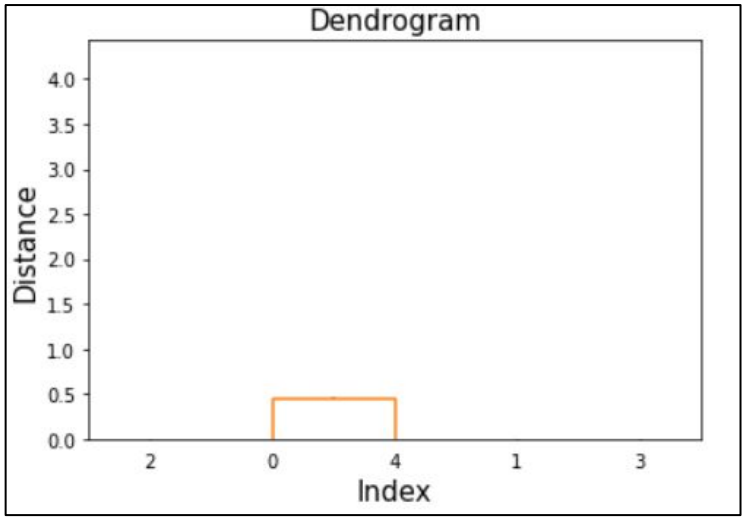
Calculate the Euclidean distance between the data points.

The distance matrix shows that the first and last points are closest.

	0	1	2	3	4
0	0	6.296824597	1.019803903	5.234500931	0.447213595
1	6.296824597	0	5.277309921	1.118033989	6.438167441
2	1.019803903	5.277309921	0	4.219004622	1.216552506
3	5.234500931	1.118033989	4.219004622	0	5.403702434
4	0.447213595	6.438167441	1.216552506	5.403702434	0

Case study





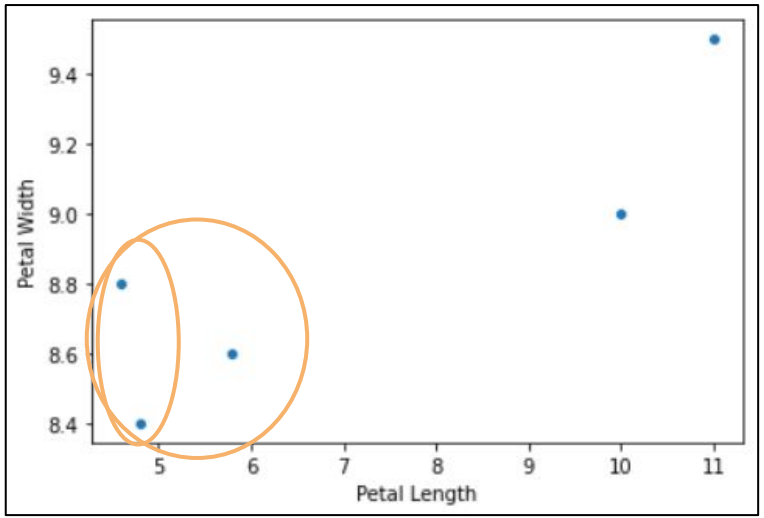
Case study

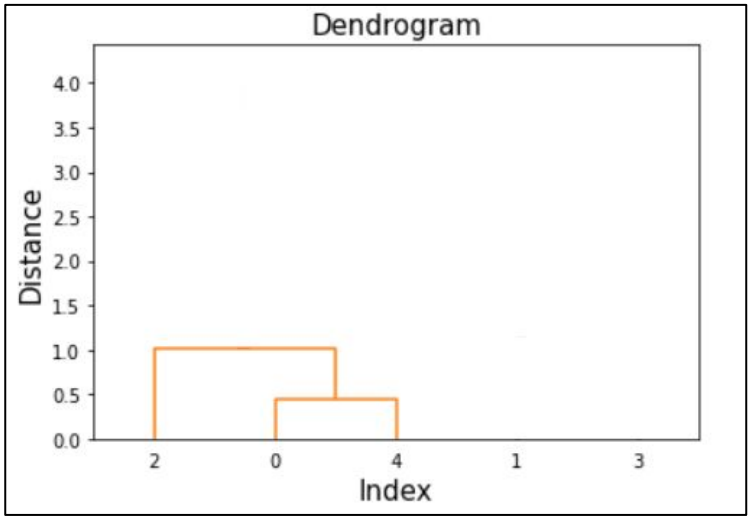
Consider (0,4) as a new cluster and calculate the distance between the cluster and data points using a single linkage.
Other distances will remain the same.

The distance matrix shows that the 3rd point is closest to the cluster (0,4).

	(0,4)	1	2	3
(0,4)	0	6.296824597	1.019803903	5.234500931
1	6.296824597	0	5.277309921	1.118033989
2	1.019803903	5.277309921	0	4.219004622
3	5.234500931	1.118033989	4.219004622	0

Case study





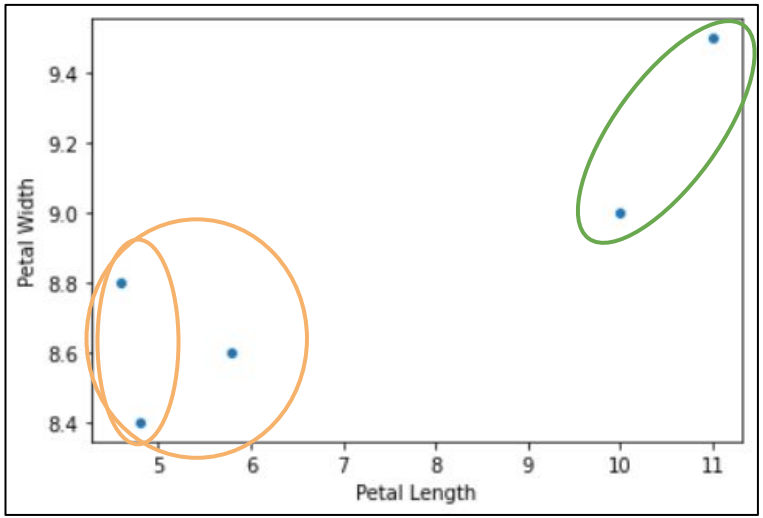
Case study

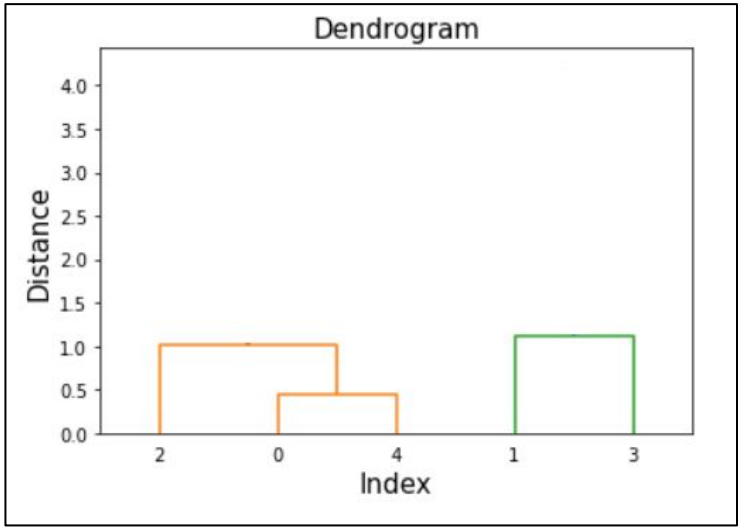
Consider ((0,4),2) as a new cluster and calculate the distance between the cluster and data points using a single linkage.
Other distances will remain the same.

The distance matrix shows that the 2nd and 4th point are closest to each other.

	((0,4),2)	1	3
((0,4),2)	0	5.277309921	4.219004622
1	5.277309921	0	1.118033989
3	4.219004622	1.118033989	0

Case study





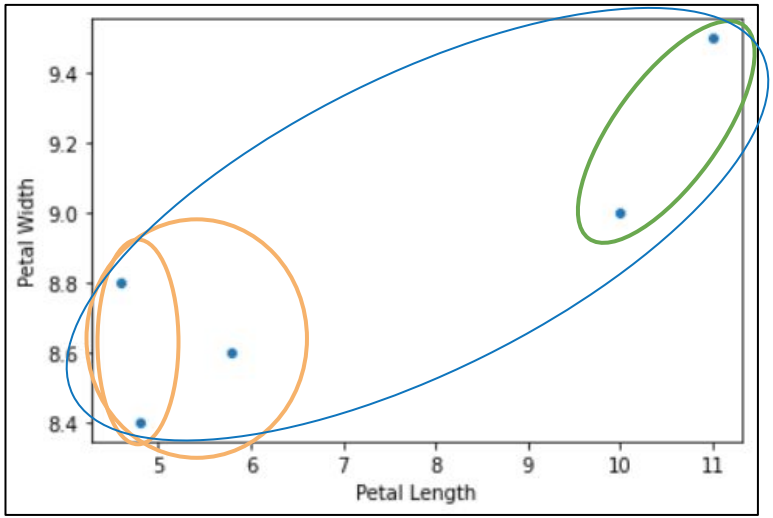
Case study

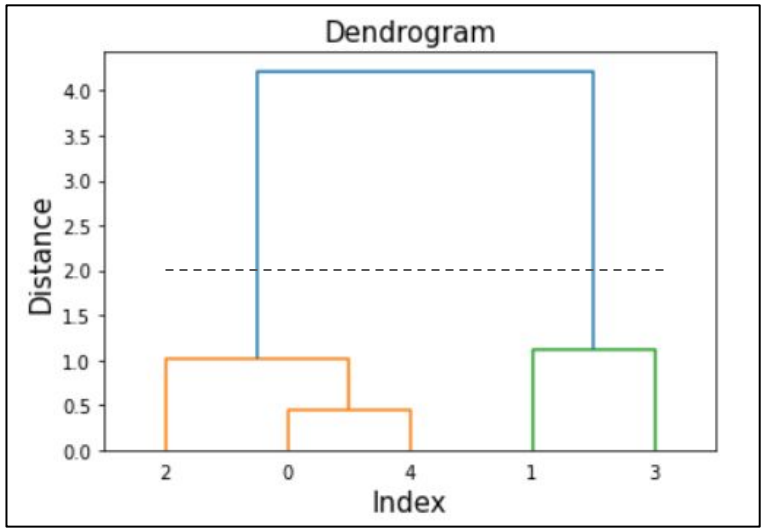
Consider (1,3) as a new cluster and calculate the distance between the clusters (1,3) and ((0,4),2) using a single linkage.

Now we have obtained the two clusters. These can be merged into a single cluster.

	((0,4),2)	(1,3)
((0,4),2)	0	4.219004622
(1,3)	4.219004622	0

Case study





If we cut the dendrogram at a height of 2.0, then we get two distinct clusters (0,2,4) and (1,3).

Case study

The cophenetic coefficient is 0.9679, which is close to 1. Thus we can say that clustering is quite good.

Cophenetic Distance	Actual Distance
4.21900462	6.2968246
1.0198039	1.0198039
4.21900462	5.23450093
0.4472136	0.4472136
4.21900462	5.27730992
1.11803399	1.11803399
4.21900462	6.43816744
4.21900462	4.21900462
1.0198039	1.21655251
4.21900462	5.40370243

Summary - clustering techniques

Great Learning
POWER AHEAD

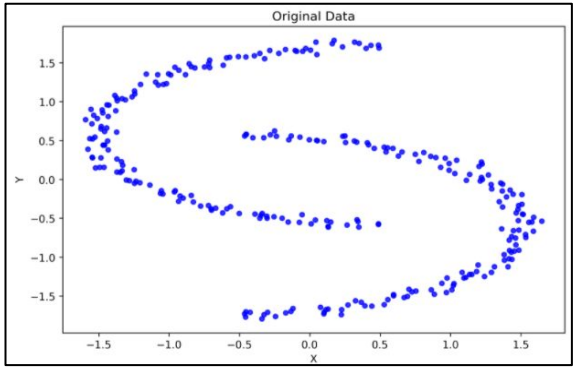
K-means

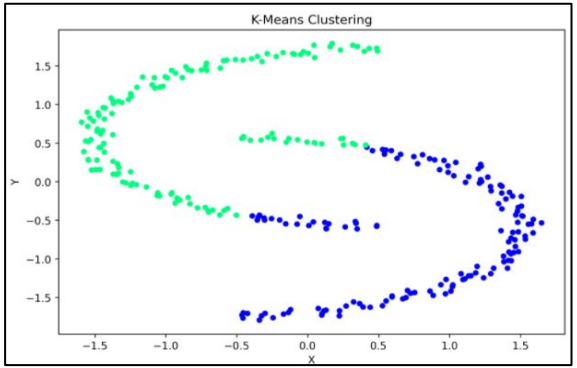
- Partition-based most widely used algorithm
- Need to specify the number of clusters (K)
- Sensitive to noise/ outliers in the data
- Creates more circular clusters
- Cluster formation depends on the initial centroid assignment

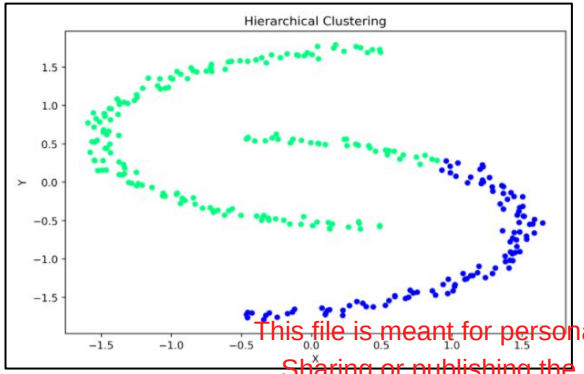
Hierarchical

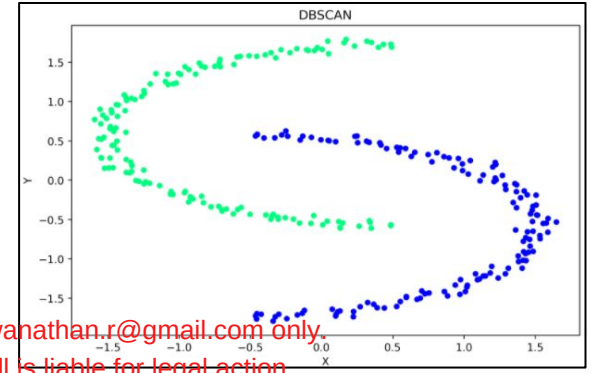
- Hierarchy-based algorithm
- No need to specify the number of clusters
- Sensitive to noise/ outliers in the data
- Useful in recovering the underlying hierarchical structure in the data
- Cluster shape depends on the distance metric

Summary - clustering techniques









Dimension Reduction

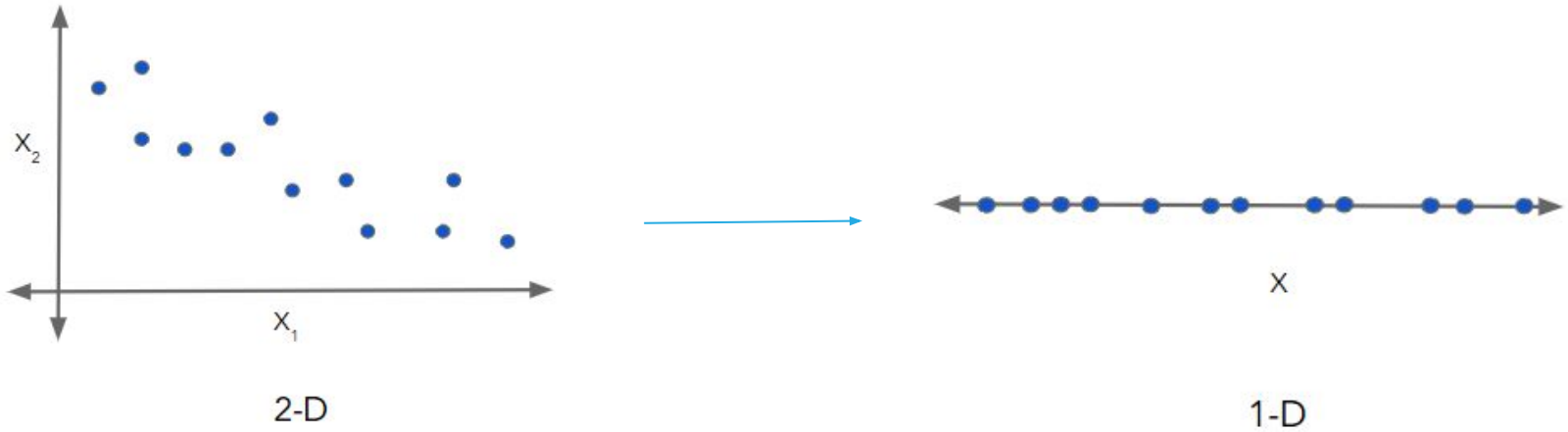
Dimension reduction

- The real-world dataset may contain a large number of features under study
- A dataset with a large number of features needs more time for model training
- As the number of variables increase, the data becomes more sparse. Overfitting can occur when a model is built on such data
- To avoid the sparsity of the data, we require more observations (i.e. rows) which may not be easily available in most cases
- The distance between the different points starts converging to a single value with an increase in dimensions. This is known as ‘Distance Concentration’

Dimension reduction

- To avoid such issues, one can reduce the dimension of the dataset
- The dimension reduction techniques remove the redundant variables/ noise in the original data, which reduces the training time
- Reducing the dataset to 2 or 3 dimensions helps in visualization of the data
- Various dimension reduction techniques:
  - Principal Component Analysis (PCA)
  - Linear Discriminant Analysis (LDA)
  - Factor Analysis

Dimension reduction



Great Learning
POWER AHEAD

Approaches for dimension reduction

- Two different approaches can be used for dimension reduction: Projection, Manifold learning
- In the projection approach, the original dataset is projected onto the lower-dimensional plane
- PCA uses the projection approach for dimension reduction
- This method is not effective if the dataset has different layers in the higher dimensions
- In manifold learning, a manifold is created on which the dataset lies

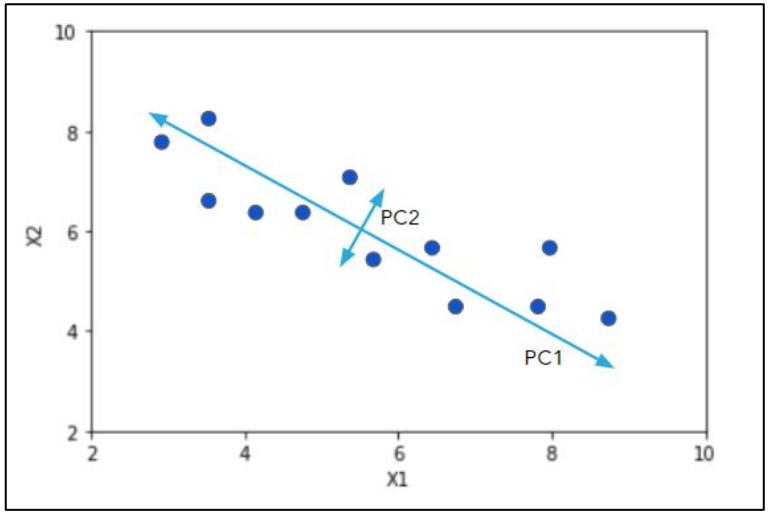
Principal Component Analysis (PCA)

PCA

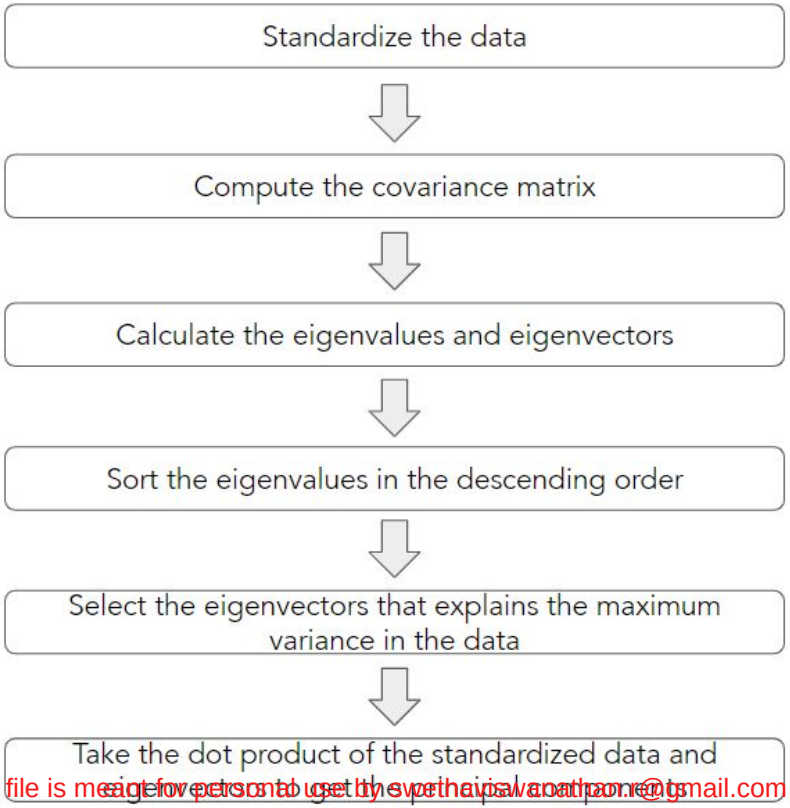
- It is one of the dimensionality reduction techniques that is used to reduce the dimensions of the large datasets
- It transforms the large set of features into a small set such that it will contain the maximum information in the original data
- The number of components is less than or equal to the number of independent variables
- PCA projects the original dataset on the lower dimensional plane
- It transforms the original data to a set of uncorrelated principal components

PCA

- The first principal component (PC1) exhibits the direction of maximum variance in the data
- It is used to remove the redundancy in the data
- PCA reduces the multicollinearity (if present) in the original data
- Principal components are always orthogonal to each other



PCA - procedure



Application

- PCA is mainly used in image compression, facial recognition models
- It is also used in the exploratory analysis to reduce the dimension of data before applying machine learning methods
- Used in the field of psychology, finance to identify the patterns high dimensional data

Great Learning
POWER AHEAD

Python code

In python, we use the following code to perform PCA:

```
# import the function
from sklearn.decomposition import PCA

# specify required no of components to 'n_components'
pca = PCA(n_components = k)

# fit_transform() fits the model and transforms the original data
# pass the standardized data to fit PCA
pca.fit_transform(standardized_data)
```

Terminologies

Covariance

- The covariance measures how co-dependent two variables are
- Positive covariance value means that the two variables are directly proportional to each other
- Negative covariance value means that the two variables are inversely proportional to each other
- It is similar to variance, but the variance illustrates the variation of the single variable and covariance explains how two variables vary together

Great Learning
POWER AHEAD

Covariance

The covariance between two variables X and Y is given by

$$COV(X,Y) = \frac{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right)}{n-1}$$

Xi = values taken by variable X $\in$ [1, n] to i $\in$ [1, n]

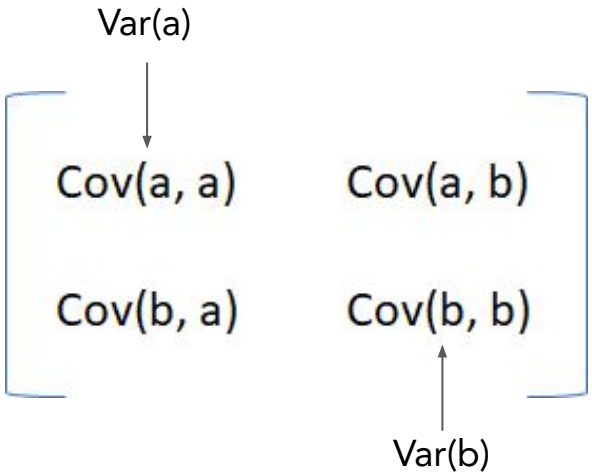
Yi = values taken by variable Y $\in$ [1, n] to i $\in$ [1, n]

$\overline{X}$ = mean of Xi

$\overline{Y}$ = mean of Yi

Covariance matrix

- The covariance matrix explains the covariance between the pair of variables
- The diagonal entries represent the variance of the variable, as it is the covariance of the variable with itself
- The diagonal matrix is always symmetric
- The off-diagonal entries are covariance between the variables that represent the distortions (redundancy) in the data



Eigenvalue

- For any nxn matrix A, we can find n eigenvalues that satisfy the characteristic equation
- A characteristic equation is defined as: $|A - \lambda I| = 0$ i.e. $\det(A - \lambda I) = 0$

  where I is the identity matrix
- The characteristic polynomial for matrix A given as $|A - \lambda I|$
- The scalar value λ is known as the eigenvalue of the matrix A
- Eigenvalues can be real/ complex in nature

Eigenvalues

- In PCA, the eigenvalue represents the total variance explained by the principal component
- The percentage of variation explained by the i^{th} component is given as

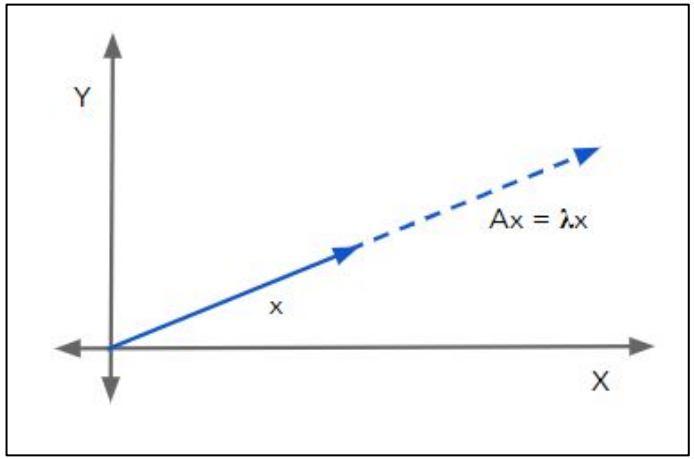
$$\left(\frac{\lambda_i}{\sum_{i=1}^{n} \lambda_i}\right) * 100$$

where λ_i is the i^{th} eigenvalue.

- The eigenvalues are considered in selecting the number of principal components

Eigenvector

- For each eigenvalue λ of a matrix A, there exist a non-zero vector x, which satisfy the equation: $(A - \lambda I)x = 0$ i.e. $Ax = \lambda x$
- The vector x is known as the eigenvector corresponding to the eigenvalue λ
- Eigenvectors of the distinct eigenvalues are always linearly independent
- The eigenvector is a vector that does not changes its direction, after transformation by matrix A



Loadings

- Mathematically, a principal component is the linear combination of the scaled (with mean = 0, standard deviation = 1) independent variables.
- i.e. PC1 = $w_{11}X_1 + w_{21}X_2 + \ldots + w_{n1}X_n$.
- The coefficients of the principal components are also known as 'loadings'
- w_1 is the loading vector consisting of elements ($w_{11}, w_{21}, \ldots, w_{n1}$) of the first principal component
- The sum of the squares of the loadings for a principal component is 1

do it YOURSELF

Great Learning
POWER AHEAD

Calculate the eigenvalues and eigenvectors of the given matrix.

$$A = \begin{bmatrix} 2 & -3 \\ 1 & 6 \end{bmatrix}$$

Selecting Principal Components

Selecting principal components

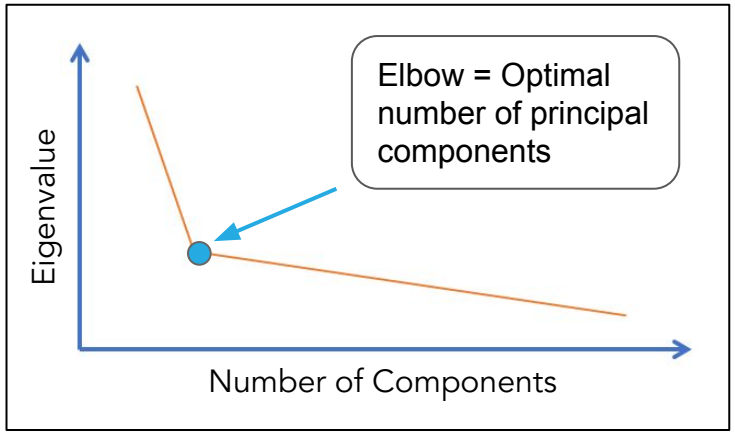
- The principal components are obtained by taking the dot product of the scaled data and the eigenvectors
- We use the eigenvalues of the covariance matrix to select the optimal number of principal components
- In order to reduce the data dimension, we consider first few principal components that explains most of the variation in the data
- Different criteria to select the principal components:
  - Kaiser Criterion
  - Scree Plot

Kaiser criterion

- It is an easy criterion to choose the principal components
- It selects the number of principal components for which the eigenvalue is greater than 1
- Higher eigenvalues explain most of the variance in the data

Scree plot

- This method plots the eigenvalues (Y-axis) against the number of principal components (X-axis)
- The elbow point in the scree plot corresponds to the optimal number of components
- After the elbow point, the components do not contribute much to the variance in the data
- This method fails if there is no explicit elbow point in the scree plot



Great Learning
POWER AHEAD

Percentage of total variance

- One can decide the number of principal components based on the percentage of variance explained by the variables
- In most of the cases, the components explaining 70-80% of the variance can be considered as the principal components
- On the other hand, in some examples, the first few components explain only 50-60% of the total variance

Summary

- PCA is an unsupervised dimension reduction technique that uses the projection method to reduce the data dimension
- It finds the principal components that best represents the data in much lower dimensions
- The first principal component explains most of the variation in the data, the second principal component explains 2nd most variation in the data and so on
- The principal components are always orthogonal to one another
- Reducing the dimension of the data to 2 or 3 dimensions aids in data visualization

Case Study

Business example

The Department of Social Welfare in Canada has collected data on the various factors that influence the crime rate (per 1,00,000 individuals) in different cities.

Three different factors are considered for the study: Total population, Unemployed individuals (between age 18-65) and Average annual income of the individual.

Let us reduce the 3-D data to 2-D to make the data more interpretable. In order to do so, we try to obtain 2 principal components which preserve the maximum information in the original data.

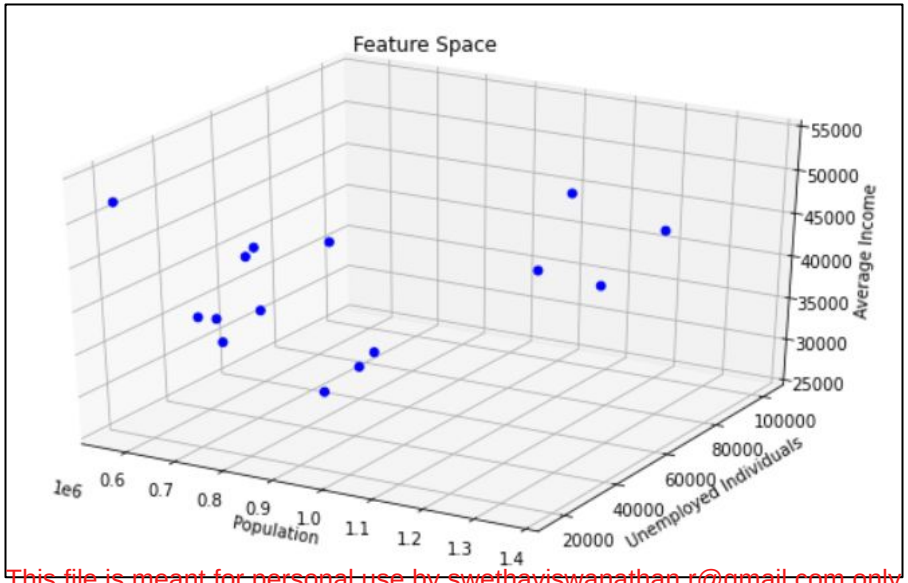
Data

Consider the independent features affecting the crime rate in Canada.

	Population	Unemployed Individuals	Average Income
0	598442	45521	31600
1	1365213	67741	48654
2	857120	36859	29800
3	685742	86100	24510
4	985303	26753	35850
5	620000	54000	41740
6	1052369	94023	46080
7	565412	16401	52100
8	674268	24758	38740
9	856200	39865	46800
10	785411	27568	40200
11	641220	36520	36305
12	1074000	102400	34000
13	654210	45214	42350
14	974100	96520	35800

Data

The original feature space in 3-D.



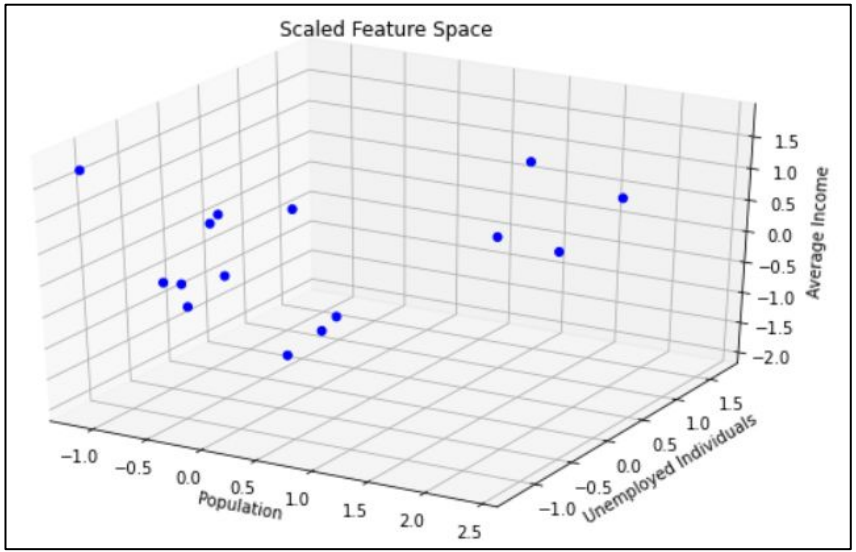
Step 1: standardize the data

$$x_{new} = \frac{x - \mu}{\sigma}$$

Where, μ: Mean of the variable

σ: Standard deviation of the variable

x: Original data points



Step 2: covariance matrix

- The diagonal entries of the matrix represents the variance of each variable
- Here, for the standardized data, the variance is 1
- The off-diagonal values exhibits the relation between pair of variables

```
[[ 1.          0.51213849  0.17461469]
 [ 0.51213849  1.         -0.23987409]
 [ 0.17461469 -0.23987409  1.        ]]
```

Step 3: eigenvalues and eigenvectors

Let A be a covariance matrix and λ be the eigenvalue of A.

Eigenvalues are the roots of the equation:

$$det(A - \lambda I) = 0$$

$$det\left(\begin{bmatrix} 1 & 0.51213849 & 0.17461469 \\ 0.51213849 & 1 & -0.23987409 \\ 0.17461469 & -0.23987409 & 1 \end{bmatrix} - \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix}\right) = 0$$

Step 3: eigenvalues and eigenvectors

After solving the equation, we get

$\lambda_1 = 0.35442161,\ \lambda_2 = 1.51704963,\ \lambda_3 = 1.12852876$

Now find the eigenvectors by solving the following equation:

$$(A - \lambda I)x = 0$$

After solving the equation, we get the eigenvectors as:

```
[[ 0.63178091 -0.68151582  0.36930891]
[-0.65513505 -0.72411768 -0.21552643]
[-0.41430778  0.10578172  0.90396863]]
```

Step 4: sort the eigenvalues

Sort the eigenvalues in the descending order.

$\lambda_2 = 1.51704963$,

$\lambda_3 = 1.12852876$,

$\lambda_1 = 0.35442161$

Since there are only 3 eigenvalues, we use the Kaiser criterion to decide the number of principal components.

Here, λ_2 and λ_3 are greater than 1. Thus we consider the eigenvectors corresponding to λ_2 and λ_3 as the loadings for principal components.

Step 5: select the eigenvectors

- Consider the eigenvectors corresponding to the eigenvalues λ_2 and λ_3

  Coefficients of the first two principal components:

```
[[-0.68151582,  0.36930891],
 [-0.72411768, -0.21552643],
 [ 0.10578172,  0.90396863]]
```

Step 6: select the components

Transform the original data by taking the dot product of the original data with the eigenvectors to get the principal components.

	PC1	PC2
0	0.775810	-1.193580
1	-1.843436	1.928682
2	0.192405	-0.923901
3	-0.605173	-2.204244
4	0.146691	0.084517
5	0.640157	-0.007702
6	-1.601667	0.915383
7	1.893958	1.425421
8	1.170228	-0.060018
9	0.357842	1.087864
10	0.786712	0.274442
11	0.939639	-0.493302
12	-2.046345	-0.558696
13	0.766898	0.186667
14	-1.573717	-0.461534

Summary